The System SOL version 2018

Makoto Hamana[1], Kentaro Kikuchi[2],

[1] Department of Computer Science, Gunma University, Japan
hamana@cs.gunma-u.ac.jp
[2] RIEC, Tohoku University, Japan
kentaro.kikuchi@riec.tohoku.ac.jp

SOL is a Haskell-based tool that assists the proofs of confluence and strong normalisation of higher-order computation. SOL is intended to be a generic higher-order computation analysis tool that is applicable to the modern theories of higher-order programming languages. This aim is demonstrated in [Ham17] and further developed in [Ham18].

Based on the foundation of second-order algebraic theories [FH10] and its computational counter part [Ham16, Ham17] and polymorphic extension [Ham18], we implemented various results on higher-order syntax and computation in SOL, including Knuth and Bendix's critical pair checking for confluence, and Function-as-Constructor Unification (FCU) [LM16] for unification. Termination analysis is based on the General Schema criterion [Bla00].

References

[Bla00] F. Blanqui. Termination and confluence of higher-order rewrite systems. In *Rewriting Techniques and Application (RTA 2000)*, LNCS 1833, pages 47–61. Springer, 2000.

[FH10] M. Fiore and C.-K. Hur. Second-order equational logic. In *Proc. of CSL'10*, LNCS 6247, pages 320–335, 2010.

[Ham16] M. Hamana. Strongly normalising cyclic data computation by iteration categories of second-order algebraic theories. In *Proc. of FSCD'16*, volume 52 of *the Leibniz International Proceedings in Informatics (LIPIcs)*, pages 21:1–21:18, 2016.

[Ham17] M. Hamana. How to prove your calculus is decidable: practical applications of second-order algebraic theories and computation. *Proceedings of the ACM on Programming Languages*, 1(1):22:1–22:28, 9 2017. Vol. Í, Issue ICFP, September 2017, Article No. 22, pp.1-28, ACM Press 2017.

[Ham18] M. Hamana. Polymorphic Rewrite Rules: Confluence, Type Inference, and Instance Validation, *Functional and Logic Programming (FLOPS'18)*, Lecture Notes in Computer Science 10818, pp.99-115, Springer, 2018.

[LM16] T. Libal and D. Miller. Functions-as-Constructors Higher-Order Unification. In *Proc. of FSCD 2016*, volume 52 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 26:1–26:17, 2016.

[Nip93] T. Nipkow. Functional unification of higher-order patterns. In *Proc. of (LICS'93)*, pages 64–74, 1993.